

## Chapter 12 Views, Indexes, and Sequences

---

---

---

---

---

---

---

---

### What Objects are Present in Your Schema?

#### ■ USER\_OBJECTS

- a Data Dictionary view (Ch 13)
- describes all objects owned by the current user
- tables, views, synonyms, sequences, indexes, functions, procedures, triggers, packages, others

```
SQL> desc USER_OBJECTS
```

```
SQL> SELECT object_name, object_type,  
         created, status  
        FROM USER_OBJECTS  
        ORDER BY object_type, object_name;
```

---

---

---

---

---

---

---

---

### View Concepts

- A stored SELECT statement
- Used to present subsets or combinations of data
- A **virtual** table based on a table or another view
  - the tables on which a view is based are called **base tables**
- Contains no data of its own
  - is like a window through which tables data can be viewed/changed
  - can't be indexed

---

---

---

---

---

---

---

---

## View Example #1

### ■ Create a View to Summarize Data

```
SQL> CREATE OR REPLACE VIEW writer_activity AS
SELECT ln || ', ' || fn author, article_count, avg_length
FROM writer INNER JOIN
    (SELECT writerid, count(*) article_count,
        avg(length) avg_length
     FROM article
     GROUP BY writerid) stats
ON writer.writerid = stats.writerid;
```

### ■ Use the View

```
SQL> desc writer_activity
SQL> SELECT * FROM writer_activity;
SQL> SELECT author, avg_length
FROM writer_activity WHERE article_count >= 3;
```

4

---

---

---

---

---

---

---

---

## Why Use Views?

### ■ To make complex queries easy

- a view is **prebuilt query**... user can run it without knowing how to:
  - use record selection criteria
  - write complex join expressions
  - derive calculated columns
  - perform sorting
  - perform grouping

### ■ To present different views of the same data

- one view to provide a detailed look at rows, others can present summaries grouped/sorted in various ways

### ■ To restrict data access

- a view can display only selected rows or columns
- privileges can be granted on the view without granting privileges on the base tables

5

---

---

---

---

---

---

---

---

## Practice Time: Using Views

```
SQL> desc article_writer
SQL> SELECT * FROM article_writer;
SQL> SELECT title, author
FROM article_writer;
SQL> set long 5000
SQL> SELECT text
FROM USER_VIEWS
WHERE view_name = 'ARTICLE_WRITER';
```

### ■ Where did the `article_writer` view come from?

- createissue25.sql script

6

---

---

---

---

---

---

---

---

## Creating a View

```
CREATE [OR REPLACE] VIEW viewname
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraintname]]
[WITH READ ONLY];
```

- OR REPLACE
  - overwrites any existing view of the same name
  - avoids having to drop the view, recreate it, and re-grant privileges
- *subquery*
  - a complete SELECT statement
  - can use aliases for the columns
  - can use joins to involve multiple tables (a [join view](#))

7

---

---

---

---

---

---

---

---

## Creating a View

```
CREATE [OR REPLACE] VIEW viewname
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraintname]]
[WITH READ ONLY];
```

- WITH CHECK OPTION
  - specifies that only rows that meet the view's WHERE clause can be inserted, updated, or deleted
  - although a view's WHERE clause restricts the rows selected when the view runs, it does not restrict DML unless you use WITH CHECK OPTION
- *constraintname*
  - is the name assigned to the CHECK OPTION constraint
- WITH READ ONLY
  - ensures that no DML operations can be performed using the view

8

---

---

---

---

---

---

---

---

## View Practice

- The `writer` table's `amount` column is considered too sensitive. Create a view named `writer_info` that shows every column from the `writer` table **except** `amount`. The view must not allow inserts, updates or deletes.
- After you've created the `writer_info` view
  - use `DESCR` to view its structure
  - use `SELECT *` to see what it produces
  - attempt to change writerid C200's amount to 300... what happens?

9

---

---

---

---

---

---

---

---

## View Example #2

```
SQL> CREATE OR REPLACE VIEW freelancers AS
  SELECT * FROM writer WHERE freelancer = 'Y'
  WITH CHECK OPTION CONSTRAINT freelancers_freelancer_ck;
```

```
SQL> SELECT * FROM freelancers;
```

```
SQL> UPDATE freelancers
  SET lastcontact = SYSDATE WHERE writerid = 'J525';
1 row updated.
```

```
SQL> INSERT INTO freelancers
  VALUES('S100', 'Smith', 'Roger', '(666) 666-6666',
         to_date('22-DEC-02'), 'Y', 0, NULL);
1 row created.
```

```
SQL> INSERT INTO freelancers
  VALUES('J543', 'Jones', 'Jane', '(777) 777-7777',
         to_date('22-DEC-02'), 'N', 0, NULL);
```

```
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation 10
```

## DML Using Views

- DML can only involve columns present in the view's definition
  - as demonstrated in the previous slide
- When use a view to insert/update/delete rows, the base table's constraints must be satisfied
- For DML, a view must not contain
  - expressions (eg: sal + 100)
  - multi-row (aggregate) functions (eg: SUM, AVG)
  - single-row functions (eg: RTRIM, LENGTH)
  - set operators (eg: INTERSECT, UNION)
  - DISTINCT
  - GROUP BY or HAVING
  - ORDER BY

```
SQL> INSERT INTO writer_activity(author, article_count, avg_length)
  VALUES('Smith, Roger', 8, 1234.56);
INSERT INTO writer_activity(author, article_count, avg_length)
  *
ERROR at line 1: ORA-01733: virtual column not allowed here 11
```

## DML on Join Views

```
SQL> CREATE VIEW article_author AS
  SELECT articlenum, title, type, length, a.writerid, ln, phone
  FROM article a, writer w
  WHERE a.writerid = w.writerid;
```

View created.

```
SQL> INSERT INTO article_author(articlenum, title, length, writerid)
  VALUES(99, 'Economy Heads South', 99, 'J525');
1 row created.
```

```
SQL> INSERT INTO article_author(ln, phone)
  VALUES('Hemingway', '(555) 555-1212');
insert into article_author(ln, phone);
  *
```

```
ERROR at line 1:
ORA-01779: cannot modify column which maps to non key-preserved table
```

- **Key-Preserved base table**
  - when the base table's primary key is unique in the **results** of the join view
  - it is not necessary that the table's key be selected for it to be key preserved
- **DML may only be performed on a key-preserved table**
  - DML may only affect a child base table (many side of the relationship) 12

## Which Columns are Updatable?

- USER\_UPDATABLE\_COLUMNS (pg 582)
  - a data dictionary view that indicates which columns can use DML

```
SQL> SELECT column_name, updatable, insertable, deletable
FROM USER_UPDATABLE_COLUMNS
WHERE table_name = 'ARTICLE_AUTHOR';
```

COLUMN_NAME	UPD	INS	DEL
ARTICLENUM	YES	YES	YES
TITLE	YES	YES	YES
TYPE	YES	YES	YES
ISSUE	YES	YES	YES
LENGTH	YES	YES	YES
WRITERID	YES	YES	YES
LN	NO	NO	NO
PHONE	NO	NO	NO

ARTICLE is a key preserved table

---

---

---

---

---

---

---

---

## Altering a View

```
ALTER VIEW viewname COMPILE;
```

- Views can't be modified
  - use CREATE OR REPLACE to redefine the view
    - egs: add another join, additional columns
- Invalid Views
  - altering/dropping a base table invalidates any views based on the table
  - the next time you use the view, Oracle attempts to revalidate the view by compiling it
- Use ALTER VIEW to explicitly **recompile** a view. Explicit recompilation lets you locate errors before run time. You may want to recompile a view after altering one of its base tables to ensure that the alteration does not affect the view and any local objects that depend on the view.

---

---

---

---

---

---

---

---

## Renaming and Dropping Views

```
RENAME oldname TO newname;
```

- Same syntax as renaming a table
- The view works the same
- Privileges to use the view remain intact
- Any dependent objects marked as invalid

```
DROP VIEW viewname;
```

- Removes the view's definition from the database
- Has no effect on the base tables
- Objects that are based on a deleted view become invalid

---

---

---


---

---

---

---

---



## Index Preliminaries

- Analogy: a textbook's index
  - Where in our book is the topic GRANT?
  - How would you find GRANT if our book had no index?
  - What if the Security chapter was moved earlier in the book?
  - What if the Security chapter was removed from the textbook?

16

---

---

---


---

---

---

---

---



## Index Concepts

- Is a schema object
- Indexes take up space
- Can significantly speed row retrieval by using ROWID pointers
  - ROWID is the unique address of each row in the database
  - the location of the row can be determined from the ROWID
- If a column is not indexed, a full **table scan** will occur
- Is automatically used by Oracle when beneficial
- Is automatically maintained by Oracle
  - Oracle must update each index when DML is performed on an indexed column
  - can slow DML operations, especially on bulk inserts, updates

17

---

---

---


---

---

---

---

---



## When Are Indexes Created?

- Automatically
  - A unique index is created automatically when you define a PRIMARY KEY or UNIQUE constraint
- Manually
  - Can create indexes on columns to speed up access to the rows
    - eg: index foreign key fields to speed join operations
    - eg: index fields that frequently appear in WHERE clauses to speed retrieval
  - Can create an index to enforce a column's uniqueness
    - Oracle recommends the unique CONSTRAINT approach

18

---

---

---

---

---

---

---

---

## Two Types of Indexes

- B-Tree
  - balanced tree
  - the default type of index
  - best for columns with high selectivity
    - i.e., when a column has lots of distinct values (eg: articlenum)
- Bitmapped
  - best for columns with low selectivity
    - i.e., when a column has few distinct values (eg: freelancer)

19

---

---

---

---

---

---

---

---

## B-Tree Indexes

- What do they look like? (page 586)
  - leaf blocks
    - one for each data value in the column, also store ROWID
  - branch blocks
    - enable quick searching
- Handout

```
SQL> SELECT ROWID, title, length FROM article;
```

- How do they help speed row retrieval?

```
SQL> SELECT title FROM article WHERE length < 1500;
```

```
SQL> SELECT COUNT(*), MAX(length) FROM article;
```

20

---

---

---

---

---

---

---

---

## Creating an Index

```
CREATE [BITMAP] INDEX index  
ON table (column[, column]...);
```

- You must have CREATE INDEX privilege
- Oracle must acquire a table-level lock before it can build the index
- eg: create an index for article table's writerid foreign key

```
SQL> CREATE INDEX article_writerid_idx  
ON article(writerid);  
Index created.
```

21

---

---

---

---

---

---

---

---

## Practice Time

- Create an index for article table's length column
- Use the following command to ensure the new index exists

```
SQL> SELECT object_name, created
       FROM USER_OBJECTS
       WHERE object_type = 'INDEX'
       ORDER BY created;
```

22

---

---

---

---

---

---

---

---

## Creating a Composite Index

```
CREATE INDEX index
ON table (column[, column]...);
```

- Can include up to 32 columns per index
- Generally, the most commonly accessed or most selective columns go first
- Create an index for Grade table's composite **foreign key** of StudentID and SectionID

```
SQL> CREATE INDEX grade_studentidsectionid_idx
       ON grade(student_id, section_id);
Index created.
```

- What is the **leading column** of this index?
- What WHERE clauses could benefit from this index?

23

---

---

---

---

---

---

---

---

## When to Create an Index

- The column is used frequently in the WHERE clause or in a join condition
  - eg: a foreign key (Oracle automatically creates an index for a pk)
- The column contains a wide range of values
- The column contains unique values
- The column contains a large number of NULL values
- The table is large and most queries are expected to retrieve less than 5-15% of its rows
  - needle in the haystack

24

---

---

---

---

---

---

---

---

## Indexes: Pros and Cons

- Pros
  - can dramatically speed up:
    - 
    -
- Cons
  - 
  -
- Which fields should you consider indexing?

25

---

---

---

---

---

---

---

---

## Rebuilding an Index

```
ALTER INDEX indexname REBUILD;
```

- Can increase index performance & storage efficiency
  - especially after much DML has occurred since index was created

```
SQL> ALTER INDEX article_writerid_idx REBUILD;  
Index altered.
```

26

---

---

---

---

---

---

---

---

## Removing an Index

```
DROP INDEX indexname;
```

- You must be the owner or have the DROP ANY INDEX privilege
- You cannot modify an index... must drop and recreate
- When a table is dropped, Oracle automatically drops all of the table's indexes

```
SQL> DROP INDEX article_writerid_idx;  
Index dropped.  
SQL> DROP INDEX grade_studentidsectionid_idx;  
Index dropped.
```

27

---

---

---

---

---

---

---

---

## Sequence Concepts

- Generates unique sequential numbers automatically
  - can be either either ascending or descending
- Is typically used to create a primary key value
- Is a sharable object
  - can be shared by different users
  - can provide values for different tables
- Sequences are independent of tables
  - can modify or drop sequence with no effect on table
  - can modify or drop table with no effect on the sequence
  - the same sequence can be used for multiple tables

28

---

---

---

---

---

---

---

---

## Creating a Sequence

```
CREATE SEQUENCE sequence
  [INCREMENT BY n]
  [START WITH n]
  [{MAXVALUE n | NOMAXVALUE}]
  [{MINVALUE n | NOMINVALUE}]
  [{CYCLE | NOCYCLE}]
  [{CACHE n | NOCACHE}];
```

- The following was in the createissue25.sql script

```
SQL> CREATE SEQUENCE articlenum_seq
  INCREMENT BY 1
  START WITH 1
  NOCACHE;
```

29

---

---

---

---

---

---

---

---

## Creating a Sequence

- CYCLE | NOCYCLE
  - specifies whether the sequence continues to generate values after reaching its MAXVALUE/MINVALUE (NOCYCLE is the default)
  - do not CYCLE if the sequence is used to generate primary key values... unless you have a reliable mechanism that purges old rows faster than the sequence cycles
  - if use NOCYCLE, once MAXVALUE limit is reached, no additional values from the sequence will be generated and you will receive an error indicating that the sequence exceeds the MAXVALUE
- CACHE n | NOCACHE
  - specifies how many values Oracle will pre-generate and keep in memory (By default, Oracle will cache 20 values)

30

---

---

---

---

---

---

---

---

## Practice Time

- Create a sequence named `evens_seq` that provides values from 100 to 200, skipping by 2.
- Use SELECT statements to generate the first 5 values from the `evens_seq` sequence
  - use /

31

---

---

---

---

---

---

---

---

## NEXTVAL and CURRVAL Pseudocolumns

- NEXTVAL
  - generates a new sequence number and places it in CURRVAL
  - you must qualify NEXTVAL with the sequence name
- CURRVAL
  - obtains the last value returned to the user's own session
    - NEXTVAL must be issued before CURRVAL contains an initial value
  - you must qualify CURRVAL with the sequence name

32

---

---

---

---

---

---

---

---

## Using a Sequence

- Sequences are generally used in INSERT and UPDATE statements
- eg: insert a new article

```
SQL> INSERT INTO article
VALUES (articlenum_seq.NEXTVAL, 'War Sucks',
       'POL', TO_DATE('01-MAY-03'),
       1894, 'J525');
```

- What articlenum was used for that new article?

```
SQL> SELECT articlenum_seq.CURRVAL
FROM dual;
```

33

---

---

---

---

---

---

---

---

## Using a Sequence

- Gaps in sequence values can occur when:
  - A ROLLBACK occurs
  - The system crashes
  - A sequence is used in another table
- But do gaps really matter?

34

---

---

---

---

---

---

---

---

## Modifying a Sequence

- You must be the owner or have the ALTER ANY SEQUENCE privilege
- Can change the increment, maximum value, minimum value, cycle option, or cache option
  - can't change the START value... the sequence must be dropped and re-created

```
SQL> ALTER SEQUENCE dept_deptno_seq
      INCREMENT BY 1
      MAXVALUE 9999
      NOCACHE
      NOCYCLE;
Sequence altered.
```

35

---

---

---

---

---

---

---

---

## Removing a Sequence

```
DROP SEQUENCE sequencename;
```

- You must be the owner of the sequence or have the DROP ANY SEQUENCE privilege
- Once removed, the sequence can no longer be referenced

```
SQL> DROP SEQUENCE evens_seq;
```

36

---

---

---

---

---

---

---

---